Simulator-independent representation of ionic conductance models with ChannelDB

David Beeman [a,1] and James M. Bower [b]

[a] *Dept. of Electrical and Computer Engineering, University of Colorado, Boulder, CO 80309*

[b] *Research Imaging Center, University of Texas Health Science Center, and Cajal Neuroscience Research Center, University of Texas, San Antonio, TX 78284*

Abstract

ChannelDB is an implementation of a database of ionic conductance models, stored in simulator-independent NeuroML format, in order to share channel models between different neural simulators. At present, ChannelDB is implemented as a stand-alone module with its own graphical user interface to the database, which is implemented with MySQL. The NeuroML development kit parser is used to create Java objects from the NeuroML format (XML) files stored in the database. These are then accessed with Java software to create simulation scripts for the particular simulator. ChannelDB with all source code and documentation may be downloaded from http://www.modelersworkspace.org.

Key words: Database; NeuroML; Realistic modeling

---

[1] Corresponding author
E-mail address: dbeeman@dogstar.colorado.edu

1 Introduction

Simulation packages such as GENESIS [1], NEURON [3], and other systems that were developed for structurally realistic neural modeling provide the means to exchange and collaborate on the development of models by providing standard environments for the construction of simulations. However, each simulator has its own scripting language for the construction of simulations, and these are sufficiently different to make the exchange of models or model components between simulators very difficult. This is because the simulation scripts are largely procedural programs that tell the simulator how to construct a model, using the tools and basic model components of the simulator, rather than being purely declarative descriptions of the model.

In order to facilitate the exchange of neuronal model descriptions in a simulator-independent format, we developed a declarative XML-based format that has become the basis for the NeuroML model description language [2]. The Modelers Workspace (MWS) project [5] is a design for a graphical environment that uses a collection of software tools to facilitate the collaborative construction of models via the WWW. It uses NeuroML as the format for the description of models and data. ChannelDB, a database of ionic conductance models, is an implementation of one of the first core components of the MWS. At present, ChannelDB is implemented as a stand-alone module, with its own graphical user interface (GUI) to the database. After further development, the ChannelDB GUI will be merged into the MWS.

This paper describes the representation used in ChannelDB. It is intended to generate discussion and collaboration for the further development and extension of this representation for a wider variety of conductance models and simulators.

2 The ChannelDB Representation

A model neuron is composed of a number of sections or compartments that contain ionic conductances that we call "channels". Here, we are using the term as used by modelers to mean the conductance arising from the behavior of many ion-conducting pores, rather than the common usage meaning an individual pore. A channel contains one or more "gates", or gating variables, with associated exponents.

As a simple example of a model to be stored in ChannelDB, consider the set of equations used to describe the Hodgkin-Huxley model of the potassium conductance found in the squid giant axon [4]. The conductance G_K of this channel is given by

$$G_K = \overline{g}_K n^4 \tag{1}$$

where $\overline{g}_K$ is the maximum conductance of the channel, and n represents the single gating variable, with an exponent of 4. The gating variable obeys the differential equation

$$\frac{dn}{dt} = \alpha_n(V)\,(1-n) - \beta_n(V)\,n \tag{2}$$

Hodgkin and Huxley fit the voltage-dependent forward rate parameter α and backward rate parameter β to the expressions

$$\alpha_n(V) = \frac{0.01(10-V)}{exp(\frac{10-V}{10}) - 1} \text{ and } \beta_n(V) = 0.125\,exp(-V/80) \tag{3}$$

A model, such as one described by Eqs. (1–3), is well suited to the hierarchical object-oriented description provided by NeuroML and by object-oriented languages such as Java. An object representing a Hodgkin-Huxley channel will have a field, or attribute, for $\overline{g}_K$ and a set of gates. A gate object will contain objects to represent the forward and backward rate parameters, and attributes for the exponent and other properties of the gate.

The appearances of a GENESIS and a NEURON script for a simulation of a conductance model based on these equations are very different. The different paradigms used by the two simulators to implement the same model make translation a more difficult problem than a simple mapping of one syntax to another. It would be very difficult to spell out a general description of a method to convert one simulation script to the form of the other.

The listing below gives an example of a NeuroML representation of this model. Unlike a simulation script, this is a declarative representation that describes everything about the model, not a procedural description of how to implement the model.

```
<neuroml class="DBChannel" author="Dave Beeman"
    description="Hodgkin-Huxley squid K channel"
    keywords="Hodgkin-Huxley potassium squid delayed rectifier"
    uniqueID="10262778758662F22@dogstar.colorado.edu"
    notes="An implemention of the GENESIS K_squid_hh channel"
    Erest="-0.07V">
    <channels>
      <channel name="K_squid_hh" class="HHChannel" permeantSpecie="K"
            Erev="0.09V" Gmax="360.0S/m^2" ivlaw="ohmic">
         <gates>
            <gate name="X" class="HHVGate" vmin="-0.1" vmax="0.05"
                timeUnit="sec" voltageUnit="V" power="4"
                instantCalculation="false" useState="false">
              <forwardRate class="ParameterizedHHRate" A="-600.0"
                   B="-10000.0" C="-1.0" D="1.0"
```

```
                  E="0.060" F="-0.01"/>
            <backwardRate class="ParameterizedHHRate" A="125.0"
                 B="0.0" C="0.0" D="1.0" E="0.07" F="-0.08"/>
         </gate>
      </gates>
      <log author="Dave Beeman" date="Jul 9, 2002 11:11:15 PM"
            literatureReference="A.L. Hodgkin and A.F. Huxley,
            J. Physiol. (Lond) 117, pp 500-544 (1952)">
      </log>
   </channel>
  </channels>
</neuroml>
```

NeuroML provides a number of templates, or classes, that may be used for this description. In this example, the K_squid_hh channel is derived from the HHChannel class, which has certain properties such as a reversal potential Erev, and a conductance density Gmax, given in units of S/m^2, which can be used to calculate $\overline{g}_K$ for a specified area of neural membrane. It also possesses a voltage activated Hodgkin-Huxley gate, derived from the HHVGate class, to represent n. The gate contains objects representing the forward and backward rate variables. The gate also contains attributes for its exponent, the range of voltages expected, the units used, and the fields "instantCalculation" and "useState". These boolean attributes, with default values of "false", provide additional flexibility for the gate objects. The former specifies that n should be set to its steady state value n_∞, and the latter is used to describe the gate in terms of the state variables $\tau = 1/(\alpha + \beta)$ and n_∞, instead of the rate variables α and β. In this example, Eqs. (3) for the rate variables are represented with a parameterized form, derived from the ParameterizedHHRate class. A TabulatedHHRate class is provided to represent the rates with with tabulated values, and an EquationHHRate may be used to represent the rate with an equation. Other types of conductances, such as calcium-dependent channels, make use of classes for concentration-dependent gates, concentration pools, and current sources.

The channel classes currently defined and implemented for ChannelDB are:

DBChannel: Wrapper class that is used to contain any channel model that is stored in ChannelDB, along with some descriptive information.

HHChannel: Class used for all the Hodgkin-Huxley type channels in the database.

HHVGate: Used as a member of the gates set of a HHChannel. It contains forward and backward rate objects that depend on voltage, as well as some additional fields to describe the behavior of the gate.

HHCGate: An ionic concentration-dependent gate, analogous to the voltage-dependent HHVGate. It provides an additional field for a reference to the object that provides the source of the ionic concentration.

HHRate: The superclass for the specialized forms for the rate variables.

ParameterizedHHRate: A subclass of HHRate that expresses rate variables in a parameterized form typical of many Hodgkin-Huxley type rate equations, $rate = (A + BV)/(C + D\exp((E + V)/F))$.

EquationHHRate: A subclass of HHRate that expresses the rate variables as equations.

TabulatedHHRate: A subclass of HHRate that allows a gate's forwardRate or backwardRate to be specified by a table at equally spaced voltage (or concentration) points.

ConcenPool: Describes a single shell model for a concentration pool, with a buildup of concentration proportional to an incoming current and a time constant for decay. The object providing the source of concentration to a HHCGate is typically formed from this class. The source of currents is provided by a set of objects of class

CurrentSource.

CurrentSource: Used by ionic concentration pools to provide information about the object that provides an ionic current.

3 The ChannelDB Implementation

The NeuroML development kit parser (from http://www.neuroml.org) is used to convert between model representations in the form of Java objects, and the NeuroML representation. Java objects are created to represent the channel description, using the classes described above. These are contained within a DBChannel object, and a single command defined in the development kit creates the NeuroML representation from the DBChannel object. In our implementation, the NeuroML representations are stored in a searchable database, which is implemented with MySQL, and accessed with the ChannelDB GUI. However, there is no requirement to use any particular database implementation or user interface, nor to even use a database at all.

After a model is selected from the database, the NeuroML parser is used to convert the representation back into a Java DBChannel object. It is then a relatively simple task to use Java string manipulation tools to produce a simulation script for the desired simulator from the information contained in fields of the DBChannel object. This conversion method has been implemented for GENESIS, and efforts are underway to extend this to NEURON.

4 Conclusion

ChannelDB is only a first step towards our vision of a distributed database of neuronal models and model components. We hope that others will join this effort to provide further enhancements to ChannelDB and to apply and extend the NeuroML representation to create other databases of channels, cells, and networks. In particular, we encourage neural modelers to represent their own channel models with ChannelDB, creating any extensions to the representation that are needed. In order to encourage this continued development, we have made all source code and documentation for ChannelDB available for downloading from http://www.modelersworkspace.org. Downloading this software will allow one to use the ChannelDB GUI to access our prototype database of channel models, and to create one's own database of channel models to share with others.

References

[1] J. M. Bower and D. Beeman, The Book of GENESIS: Exploring Realistic Neural Models with the GEneral NEural SImulation System, second edition. (Springer-Verlag, New York, 1998).

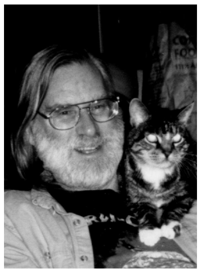
[2] N. Goddard, M. Hucka, F. Howell, H. Cornelis, K. Shanka, and D. Beeman, Towards NeuroML: Model Description Methods for Collaborative Modelling in Neuroscience, Philos. Trans. Roy. Soc. 356 (2001) 1209–1228.

[3] M Hines and N. T. Carnevale, The NEURON Simulation Environment, Neural Computation 9 (1997) 1179–1209.

[4] A. Hodgkin and A. Huxley, A quantitative description of membrane current and its application to conduction and excitation in nerve, J. Physiol. (London) 117 (1952) 500–

544.

[5] M. Hucka, K. Shankar, D. Beeman, and J. M. Bower, The Modeler's Workspace: Making model-based studies of the nervous system more accessible, in G. Ascoli, ed., Computational Neuroanatomy: Principles and Methods, (Humana Press, Totowa NJ, 2002).



David Beeman is Professor Adjunct of Electrical and Computer Engineering at the University of Colorado, Boulder, where he is developing educational materials for computational neuroscience, using the GENESIS simulator. Dr. Beeman also participates in the Modeler's Workspace project to create a software framework for the storage, editing, and running of neural models that are stored in a heterogenous network of databases. Previously, he spent 20 years at Harvey Mudd College engaged in undergraduate teaching and research in computational solid state physics, after receiving his Ph.D. in theoretical solid state physics from UCLA in 1967.

James M. Bower received his Ph.D. in neurophysiology from the University of Wisconsin-Madison. After a postdoctoral fellowship at NYU and the Marine Biological Laboratory in Woods Hole, Dr. Bower was a professor at the California Institute of Technology for 17 years. In 2002, he moved to a joint position as Professor of Computational Biology at the University of Texas Health Science Center in San Antonio, and the University of Texas San Antonio. Dr. Bower's research is focused on both the cerebellum and the mammalian olfactory system, and his laboratory has been involved in numerous science infrastructure projects including GENESIS. Dr. Bower also has a long-standing interest and involvement in early science education.